AD-A259 438

(12)

NAVSWC TR 91-358

NONNUMERIC PATTERN CLASSIFIERS

BY L. R. ELKINS, A. FARSAIE, AND D. VALDEZ

WEAPON SYSTEMS DEPARTMENT
AND
ENGINEERING AND INFORMATION SYSTEMS DEPARTMENT

JUNE 1991

Approved for public release; distribution is unlimited.

DTIC
ELECTE
JAN22 1993
S E D

NAVAL SURFACE WARFARE CENTER

Dahlgren, Virginia 22448-5000 • Silver Spring, Maryland 20903-5000

93 1 21

93-01083

NAVSWC TR 91-358

NONNUMERIC PATTERN CLASSIFIERS

BY L. R. ELKINS, A. FARSAIE, AND D. VALDEZ
WEAPONS SYSTEMS DEPARTMENT
AND
ENGINEERING AND INFORMATION SYSTEMS DEPARTMENT

JUNE 1991

Approved for public release; distribution is unlimited.

NAVAL SURFACE WARFARE CENTER
Dahlgren, Virginia 22448-5000 • Silver Spring, Maryland 20903-5000

FOREWORD

Artificial neural systems offer promising programming paradigms for solving complex problems in the area of perception, pattern recognition, and adaptive behavior. This technical report describes the application of such paradigms to the field of knowledge processing and acquisition, focusing on the relevant task of machine learning.

This study was supported by the Independent Research program office.

This technical report was reviewed by Mr. Kenneth F. Caudle, Head of the Advanced Weapons Division.

Approved by:

David S. Malyevac

DAVID S. MALYEVAC, Deputy Dept. Head
Weapons Systems Department

INSPECTED 5

Accesion For	
NTIS CRA&I	X
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution /	
Availability Codes	
Dist	Avail and / or Special
A-1	

ABSTRACT

Theoretical study was performed to determine if artificial neural systems can be used to generalize rules from examples. A method for generating rules from a multi-layer network was investigated.

When the network was presented with few training patterns, the rules derived from the network classified all patterns correctly. The generalization capabilities of the Pao-Hu and neural network classifiers were compared with each other and with the ID3 method. Classifiers were compared by applying all methods to several data sets and examining the similarities and differences among them. It was demonstrated that the neural network could act as a rule generator for an expert system. The tests have shown that the network can correctly generate rules and subsequently correctly classify patterns.

CONTENTS

ILLUSTRATIONS

TABLES

INTRODUCTION

One of the most time consuming tasks in building an expert system is the formulation of rules obtained by interviewing human experts, otherwise known as knowledge engineering. As an alternative to the task of using a knowledge engineer to bridge this gap between the expert's knowledge and the formulated rules of the expert system, an Artificial Neural System (ANS) may be used. Since ANSs are good at generalization, given a set of examples, the time needed to generate the rules of the system may be drastically reduced, as well as the number of rules actually needed to cover all of the possible input cases. By using an artificial neural system to generate the rules for the system, the task of knowledge engineering may not only be automated but optimized. The classes of expert systems that use this method of rule generation are connectionist expert systems.

There are several conventional approaches for rule generation from a body of labeled examples. Among these are the ID3 (Quinlan, 1983)[1] and Pao-Hu (Pao, 1989)[2] methods. The ID3 method creates an efficient decision tree for classifying patterns consisting of nonnumeric feature values. The data necessary to create such a tree is a set of patterns, each described by a set of nonnumeric features and a class label. This data is called the 'training set' for the procedure. The training set should consist of a subset of all of the possible patterns. The goal is to discover from the training set what minimum combinations of feature values are necessary to determine class memberships for patterns. The Pao-Hu method creates a set of rules, rather than a decision tree, to classify the patterns. As in the ID3 approach, it uses a set of labeled example patterns for its input, however, an inferencing technique is used to discover the conditions that describe class membership.

Classification trees are popular due to their conceptual simplicity and their computational efficiency. A large variety of methods have been proposed for the design of classification trees, and these methods have been successfully applied to a diverse set of problems. To construct a classification tree, it is assumed that a data set consisting of feature vectors and their corresponding class labels is available. The classification tree is then constructed by recursively generating the tree in such a way as to recursively partition the feature space.

ANS algorithms such as backpropagation networks have attracted

a great deal of interest because of their apparent potential for learning complex pattern recognition tasks. Feedforward layered neural networks are being used in a wide variety of applications in computer vision and pattern recognition. These networks consist of multiple hidden layers. Generally, these networks have no built-in hierarchy and consequently are fully connected. A related architecture has been employed by Allen (1988)[3] for modeling a variety of high-level cognitive processes. These architectures include a 3-layer backpropagation network, with an additional layer of neurons, which maintains previous internal states of the network. Unlike traditional state machines, these systems are self-adaptive. Training such systems in stages can produce effective learning. Presumably, the use of an adaptive training schedule allows a simple representation to be developed initially. The initial representation then allows the network to learn longer and more complex patterns.

Artificial neural systems exhibit characteristics of associative memory, pattern matching, generalization, and learning by example (Rumelhart, 1986)[4]. Insofar as they directly compute specific outputs in response to specific inputs, ANS can be viewed as implementing reflexive task knowledge. Knowledge based systems have demonstrated the ability to encode and exercise expert knowledge within limited domains, providing a hierarchical software organization and explainable solutions. The inferencing capabilities of these systems effectively implement declarative task knowledge.

There are important advantages to constructing rule-based systems using artificial neural systems (Gallant, 1988)[5]. One is time savings for expert system development and the potential of using large knowledge bases. The neural network is used as a design aid for the expert himself. The tool generalizes from the trainer's examples to rules it forms itself. In the case where a database exists, the net uses that source for the training examples for learning. The rules are tested by the expert system shell on sample problems. Results reflect where more training is required.

DATA PREPARATION

This technical report focuses on methods for classifying patterns with nonnumeric features. A single nonnumeric feature represents a characteristic of an object expressed in a symbolic rather than numeric fashion. An object might have a nonnumeric

feature reflecting a color, with the possible values blue or brown.

The data used for experiments was originally taken from Pao (1989)[2]. The data follows the format described below. The first line of the input file contains a list of nonnumeric feature names, separated by commas. Following this header will be a set of patterns, one per line. Each pattern contains a value for each of the features from the header line, separated by commas. The locations of these feature values in each pattern correspond to the locations of the features in the header line. Each pattern will also contain a class label. The class label will be the last component of the pattern, separated from the feature values by a colon.

As more data was needed for purposes of experimentation, it was constructed in the same form. Four datasets are used in this report. Each contains a number of patterns, comprised of four non-numeric features and a class label. The class of each pattern may be either "a" or "o". Classes have no physical meaning, other than to indicate a logical category.

The first dataset, Dataset 1, contains a set of twenty-four patterns, representing all possible combinations of the given features (Table 1). Of these twenty-four patterns, seventeen are of class "o", and seven of class "a". Dataset 2 (Table 2) is a subset of Dataset 1 containing only thirteen of the twenty-four patterns. Of these, seven are of class "o", and six of class "a". Dataset 3 (Table 3) is another subset of Dataset 1, containing five patterns of class "o", and only two of class "a". Dataset 4 (Table 4) is again a subset of Dataset 1, and contains eight patterns of class "o" and four of class "a". Dataset 4 is used primarily for in-depth examples of the Pao-Hu method and rule derivation.

TABLE 1. DATASET 1

	WEIGHT	HEIGHT	HAIR	EYES	CLASS
1.	light,	tall,	dark,	brown:	o
2.	light,	short,	dark,	brown:	o
3.	light,	tall,	dark,	blue:	o
4.	light,	short,	dark,	blue:	o
5.	light,	tall,	blond,	brown:	o
6.	light,	short,	blond,	brown:	o
7.	light,	tall,	blond,	blue:	o
8.	light,	short,	blond,	blue:	o
9.	light,	tall,	red,	brown:	o
10.	light,	short,	red,	brown:	o
11.	light,	tall,	red,	blue:	o
12.	light,	short,	red,	blue:	o
13.	heavy,	tall,	dark,	brown:	a
14.	heavy,	short,	dark,	brown:	o
15.	heavy,	tall,	dark,	blue:	o
16.	heavy,	short,	dark,	blue:	o
17.	heavy,	tall,	blond,	brown:	o
18.	heavy,	short,	blond,	brown:	a
19.	heavy,	tall,	blond,	blue:	o
20.	heavy,	short,	blond,	blue:	a
21.	heavy,	tall,	red,	brown:	a
22.	heavy,	short,	red,	brown:	a
23.	heavy,	tall,	red,	blue:	a

TABLE 2. DATASET 2 (SUBSET OF DATASET 1)

	WEIGHT	HEIGHT	HAIR	EYES	CLASS
1.	light,	tall,	dark,	brown:	o
10.	light,	short,	red,	brown:	o
11.	light,	tall,	red,	blue:	o
12.	light,	short,	red,	blue:	o
13.	heavy,	tall,	dark,	brown:	a
14.	heavy,	short,	dark,	brown:	o
15.	heavy,	tall,	dark,	blue:	o
17.	heavy,	tall,	blond,	brown:	o
18.	heavy,	short,	blond,	brown:	a
20.	heavy,	short,	blond,	blue:	a
21.	heavy,	tall,	red,	brown:	a
23.	heavy,	tall,	red,	blue:	a
24.	heavy,	short,	red,	blue:	a

TABLE 3. DATASET 3 (SUBSET OF DATASET 1)

	WEIGHT	HEIGHT	HAIR	EYES	CLASS
2.	light,	short,	dark,	brown:	o
9.	light,	tall,	red,	brown:	o
10.	light,	short,	red,	brown:	o
14.	heavy,	short,	dark,	brown:	o
17.	heavy,	tall,	blond,	brown:	o
18.	heavy,	short,	blond,	brown:	**a**
21.	heavy,	tall,	red,	brown:	**a**

TABLE 4. DATASET 4 (SUBSET OF DATASET 1)

	WEIGHT	HEIGHT	HAIR	EYES	CLASS
1.	light,	tall,	dark,	brown:	o
2.	light,	short,	dark,	brown:	o
8.	light,	short,	blond,	blue:	o
9.	light,	tall,	red	brown:	o
10.	light,	short,	red,	brown:	o
12.	light,	short,	red,	blue:	o
14.	heavy,	short,	dark,	brown:	o
17.	heavy,	tall,	blond,	brown:	o
18.	heavy,	short,	blond,	brown:	a
21.	heavy,	tall,	red,	brown:	a
22.	heavy,	short,	red,	brown:	a
24.	heavy,	short,	red,	blue:	a

RULE-BASED CLASSIFIERS

There are two conventional algorithms of interest for classifying patterns with nonnumeric feature values. The first is the ID3 approach, by Quinlan (1983)[1]. This algorithm creates an efficient decision tree to classify the patterns. It uses labeled example patterns to determine how the features may be examined in sequence, until all of the labeled example patterns have been classified correctly. The second algorithm is the Pao-Hu method (Pao, 1989)[2]. This algorithm creates a set of rules, rather than a decision tree, to classify the patterns. The following section will describe these two methods in detail.

ID3 METHOD

Concept

The ID3 algorithm (Quinlan, 1983)[1] for classification of patterns with nonnumeric features was implemented. This procedure creates an efficient decision tree for classifying patterns consisting of nonnumeric feature values. The data necessary to create such a tree is a set of such patterns, each with a labeled class. This data is called the 'training set' for the procedure. The training set should consist of a subset of all possible patterns given the combinations of features and their values available. The goal is to discover, from the training set, what minimum combinations of feature values are necessary to determine class memberships for patterns, both those that are in the training set (with their labeled classes) and those that are not. Each pattern to be classified has the same number of features but different values associated with those features.

The primary equation used by ID3 to determine how the decision tree will be created is the entropy equation (Pao, 1989)[2], or

$$\text{Entropy} = -p\log_2(p)$$

where p is defined as probability, based on frequency of occurrence. The smaller the entropy, the greater the amount of information present. The procedure is that at any point in the creation of the tree, an entropy value is calculated for a feature (that is not already used in the tree). The smaller the entropy value calculated, the greater the amount of information gained if that feature is used in the next level of the tree.

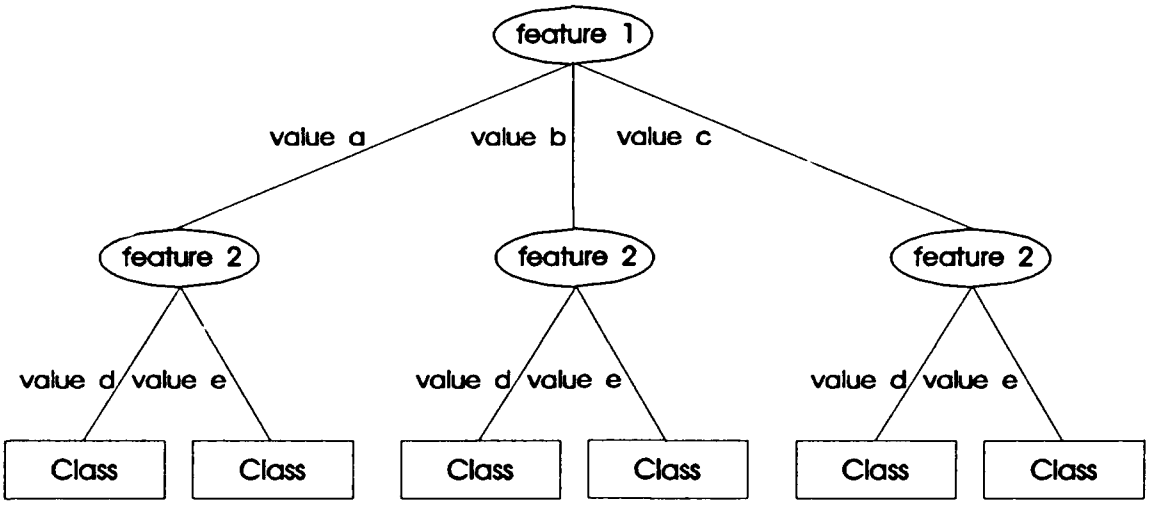

FIGURE 1. FORM OF A COMPLETED DECISION TREE

The completed decision tree will be of the form shown in Figure 1, where the number of values (and therefore the number of branches) for each feature will vary. The number of features may also vary, depending on the problem.

Algorithm

The algorithm to create a decision tree from a set of labeled examples using the ID3 method is as follows.

1. First calculate the initial entropy of the system with the following equation,

$$\text{Initial System Entropy} = \sum_{i=1}^{C} -p(i)\log_2(p(i))$$

where C equals the total number of classes found in the training set, and p(i) equals the probability of the ith-type class. The probability p(i) is actually a ratio of the form

$$p(i) = \frac{\text{number of patterns within class i}}{\text{total number of patterns in the training set}}$$

From here on, the number of patterns in class i will be referred to as c(i), and the total number of patterns in the

training set will be referred to as P.

For example, given 8 patterns in the training set and 2 different kinds of classes, the initial system entropy would be

$$\text{Initial System Entropy} = -\frac{c(1)}{8}\log_2\left(\frac{c(1)}{8}\right) - \frac{c(2)}{8}\log_2\left(\frac{c(2)}{8}\right)$$

2. The next step is to determine which feature, in the list of different features found from the training set, will be used as the root of the decision tree.

   a. For each feature k(i), i=1,2,...,K, divide the number of patterns in the training set, P, into categories corresponding to its values. Do this by looking at each pattern to see what value it has for the feature k(i) and put the pattern in the appropriate value branch. For the algorithm the feature k(i) corresponds to the root of the tree, and its values correspond to the branches of the root. For J different values of the feature k(i), refer to the set of patterns with value(j) for feature k(i) as n(j). Here, n(j) is the branch population for the value j, where j=1,2,...,J.

   b. After all of the patterns have been sorted to their appropriate branches, calculate the entropy for each branch with the following equation.

$$\text{Branch Entropy } (k(i),j) = \sum_{m=1}^{C} \frac{n(j,m)}{n(j)}\log_2\frac{n(j,m)}{n(j)}$$

   where n(j), as stated above, is the branch population for the value j (for feature k(i)), and n(j,m) is the number of patterns with class m in the jth branch.

   c. The feature entropy for k(i) may then be calculated by summing all of the branch entropies. The equation for feature entropy is the following.

$$\text{Feature Entropy } k(i) = \sum_{j=1}^{J} \frac{n(j)}{P} * [\text{Branch Entropy } (k(i),j)]$$

   d. Find the feature for which the smallest feature entropy value was calculated. Use the feature (from step d) as the root of the decision tree. Once the root, with its value branches, has been created as the first level of the decision

tree, the new system entropy will be the feature entropy value that was calculated for the root feature.

3. If the new system entropy is not 0, then find one of the other features (that has not been built into the decision tree) to use in the next level of the decision tree. This step is performed in the same manner that rootnode was performed. But the fact that the tree is now partially created must be taken into account.

   a. For each feature k(i), i=1,2,...,K, not already built into the decision tree, find the feature entropy based on the partially created tree.

      i. For each value branch of the most recently added feature of the tree, divide the number of patterns into categories corresponding to k(i)'s feature values. This is similar to step 2(a) above, but the process is done for each previous value branch.

         The most recently added feature of the tree will be referred to as prevk, and the values of that feature will be referred to as prevk(m), m=1,2,...,M, where M is the number of values (or value branches) associated with prevk.

         Therefore, step i can be written as

            For each m, m=1,2,...,M, divide the number of patterns into categories corresponding to k(i)'s feature values.

      ii. For each prevk(m), and for each value of k(i), find the branch entropy using the equation in step 2(b). In other words, if j=1,2,...,J, where J is the number of different values associated with k(i), then

         For each m, m=1,2,...,M, and for each j, j=1,2,...,J, find Branch Entropy (k(i),j).

      iii. For each prevk(m), calculate the feature entropy using the equation in step 2(c). So

         For each m, m=1,2,...,M, find Feature Entropy (k(i),m).

      iv. The population (or number of patterns) of each prevk(m) will be referred to as prevkn(m). Now the total feature entropy can be found (based on the partially created decision tree) using the following formula.

   b. After the completion of 3(a), find the smallest total

$$\text{Total Feature Entropy } k(i) = \sum_{m=1}^{M} \frac{\text{prevkn}(m)}{P} * [\text{Feature Entropy } (k(i), m)]$$

feature entropy value.

c. Choose the corresponding feature k(i) with that entropy value to be built into the next level of the decision tree. The feature, with its value branches, will need to be attached to a maximum of M value branches, which are the value branches associated with prevk. Once this feature has been incorporated into the decision tree, the new system entropy will be the smallest total feature entropy that was calculated for the chosen feature.

4. Repeat step 3, until either the system entropy is 0 or until all of the features have been built into the tree.

Experimental Results

Consider Dataset 1, shown in Table 1, as an entire test set. The tree generated by the ID3 method from the data in Dataset 1 is shown in Figure 2. This decision tree correctly identified all examples from this dataset.

Next, a subset of the original file is considered, in the form of Dataset 2 (Table 2). The result of the ID3 algorithm on this dataset is shown in Figure 3. This tree has the same decisions and conclusions as that of Figure 2. The examples removed from Dataset 1 to create Dataset 2 did not affect the tree.

Next, ID3 was applied to Dataset 3 (Table 3). The results are shown in Figure 4. The tree produced is similar with the exception of one branch. Those patterns having WEIGHT=heavy are queried on their hair color. In the tree produced from Dataset 3, the conclusion is that those with HAIR=dark are of class o. Since the pattern (heavy, dark, tall, brown) in Dataset 1 is of class a, this is incorrect. Examination of Dataset 3 shows that this pattern is not present in those used to build this tree, while (heavy, short, dark, brown) of class o was. The algorithm picked the best tree on the basis of the information it had, misclassifying one out of the original twenty-four patterns.

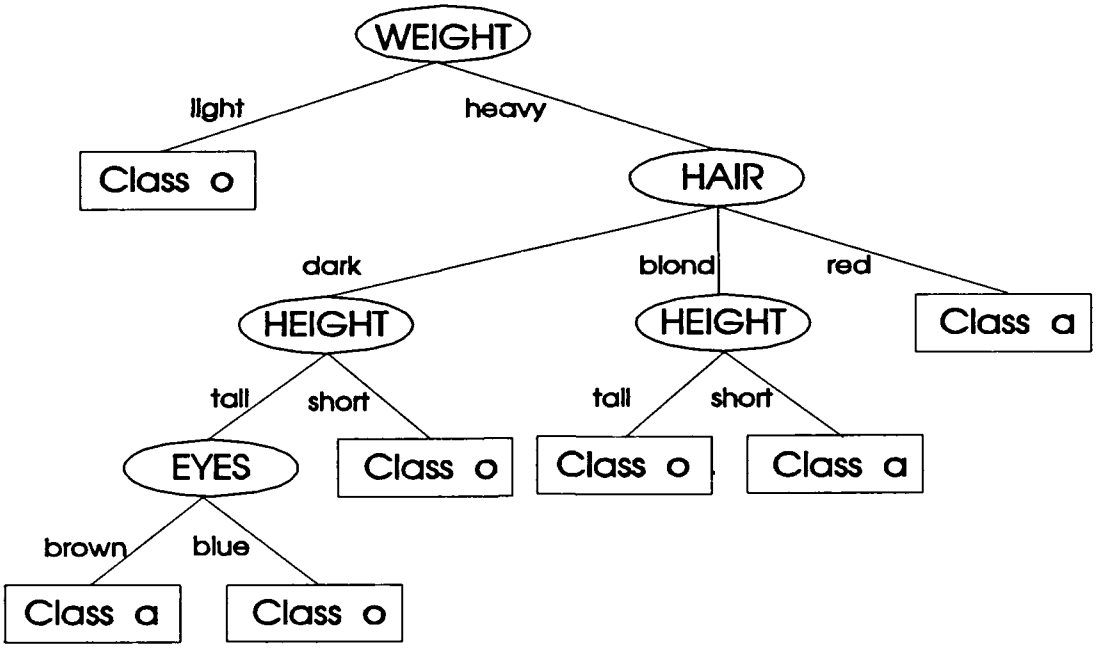

FIGURE 2. DECISION TREE GENERATED FROM DATASET 1

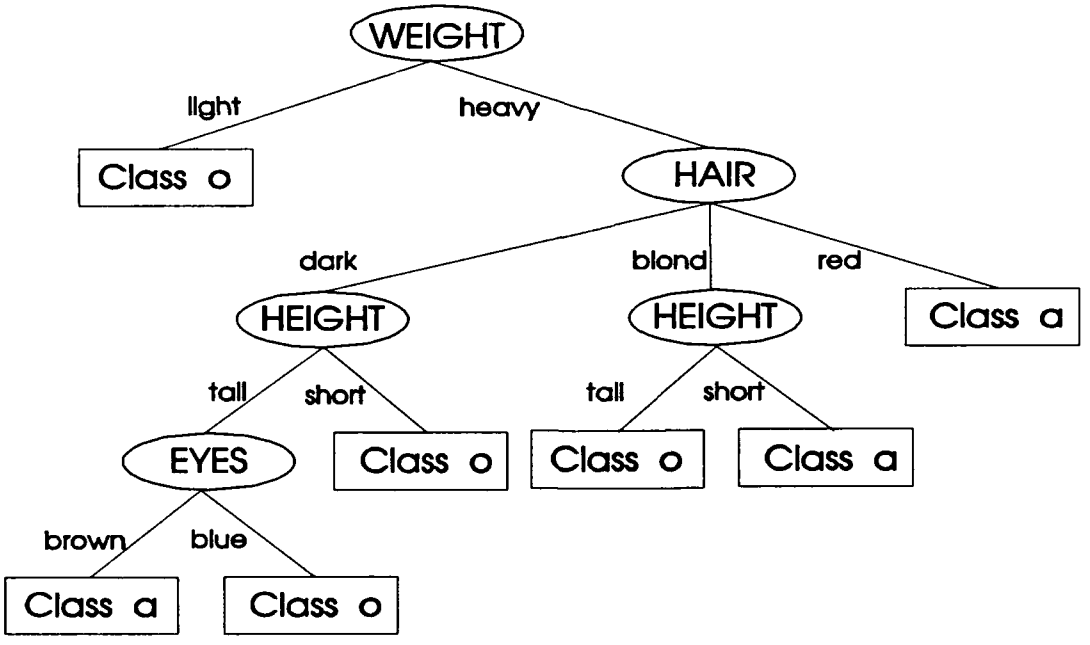

FIGURE 3. DECISION TREE GENERATED FROM DATASET 2

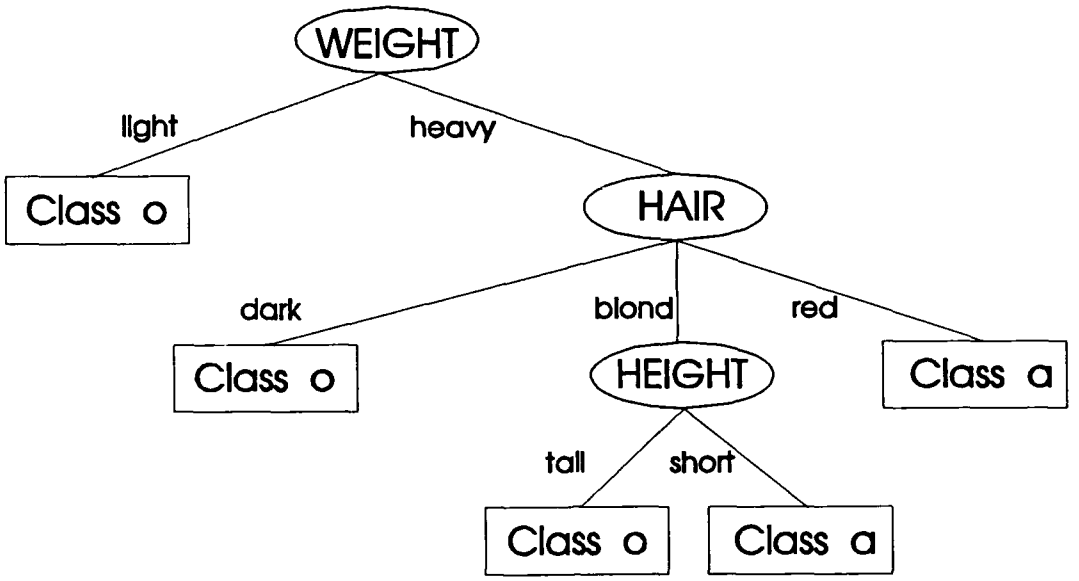

FIGURE 4. DECISION TREE GENERATED FROM DATASET 3

PAO-HU METHOD

Concept

The Pao-Hu method uses list processing techniques to infer a minimal set of classification rules sufficient to classify non-numeric patterns. The Pao-Hu method produces rules in a Sum Of Products form (several AND clauses joined by an OR connective), with one rule for each class. Each individual AND clause is referred to as a partial rule, and the entire expression formed by joining all partial rules for a given class with an OR connective is referred to as the complete rule. Because the partial rules are ORed together, faster evaluation of the entire expression can be performed (if any partial rule evaluates to TRUE, then the complete rule evaluates to TRUE).

Rules are formed by considering the values of features, referred to as descriptors, contained in the patterns. The Pao-Hu method creates partial rules that are as short as possible by using only as many descriptors as are necessary to determine class membership. Each partial rule will then produce as few queries about the features of the current pattern as necessary.

To generate the partial rules, all descriptors are first considered individually to see if they uniquely identify a class. If any do, they are considered to be distinct descriptors (or first-level descriptors). All distinct descriptors form partial rules. (WEIGHT=light) is an example. When all single descriptors

are examined, the search continues with the conjunction (the AND) of two descriptors. If this conjunction uniquely identifies a class, second level descriptors are constructed. For example, (WEIGHT=heavy)&(HAIR=red) is a second level descriptor. This continues until all of the example patterns are covered by a partial rule. The OR of all of the partial rules will result in the complete rule.

Because the partial rules are ORred together, evaluation of the entire expression can be performed (if any partial rule evaluates to TRUE, then the complete rule evaluates to TRUE). Since each of the partial rules are short (requiring a minimum of queries to be evaluated) and independent (evaluation of one partial rule has no bearing on another), the rules produced by the Pao-Hu method are well suited for use on a parallel machine. Each attempt to evaluate a partial rule can be treated as a process. When any process determines that its partial rule has fired, the related complete rule can be fired.

Algorithm

The Pao-Hu method uses list processing techniques to infer classification rules.

First, the patterns are rewritten in terms of their descriptors. For the data in Dataset 4 (Table 4), this gives us the following (Table 5).

TABLE 5. DATASET 4 EXPRESSED AS CONJUNCTIONS OF DESCRIPTORS

Pattern Number	Pattern Description	Class Label
1	(WEIGHT=light)&(HEIGHT=tall)&(HAIR=dark)&(EYES=brown)	o
2	(WEIGHT=light)&(HEIGHT=short)&(HAIR=dark)&(EYES=brown)	o
3	(WEIGHT=light)&(HEIGHT=short)&(HAIR=blond)&(EYES=blue)	o
4	(WEIGHT=heavy)&(HEIGHT=tall)&(HAIR=blond)&(EYES=brown)	o
5	(WEIGHT=heavy)&(HEIGHT=short)&(HAIR=dark)&(EYES=brown)	o
6	(WEIGHT=heavy)&(HEIGHT=short)&(HAIR=blond)&(EYES=brown)	a
7	(WEIGHT=light)&(HEIGHT=tall)&(HAIR=red)&(EYES=brown)	o
8	(WEIGHT=light)&(HEIGHT=short)&(HAIR=red)&(EYES=blue)	o
9	(WEIGHT=light)&(HEIGHT=short)&(HAIR=red)&(EYES=brown)	o
10	(WEIGHT=heavy)&(HEIGHT=tall)&(HAIR=red)&(EYES=brown)	a
11	(WEIGHT=heavy)&(HEIGHT=short)&(HAIR=red)&(EYES=blue)	a
12	(WEIGHT=heavy)&(HEIGHT=short)&(HAIR=red)&(EYES=brown)	a

Second, a table of descriptors and their associated patterns is formed (Table 6).

TABLE 6. DESCRIPTORS AND THEIR PATTERNS

Descriptor	Associated Patterns Class o	Class a
WEIGHT=light	1,2,3,7,8,9	
WEIGHT=heavy	4,5	6,10,11,12
HEIGHT=tall	1,4,7	10
HEIGHT=short	2,3,5,8,9	6,11,12
HAIR=dark	1,2,5	
HAIR=red	7,8,9	10,11,12
HAIR=blond	3,4	6
EYES=brown	1,2,4,5,7,9	6,10,12
EYES=blue	3,8	11

One class is selected for rule generation (the target class). The first partial rules are formed by using distinct descriptors. Distinct descriptors are descriptors whose patterns are present in only the target class. Since at this point descriptors are considered individually, they are referred to as first level descriptors. For class o, the first level descriptors are (WEIGHT=light) and (HAIR=dark), since these descriptors are found in patterns belonging to the target class, and not to any other class. Therefore,

pattern $\in$ class o if it contains (WEIGHT=light), or
pattern $\in$ class o if it contains (HAIR=dark).

After all first level distinct descriptors are considered, then the distinct descriptors used are removed from the list. Since any patterns containing these descriptors can be classified by the information already obtained, all patterns covered by the distinct descriptors can be removed. The resulting table contains a list of patterns which have not yet been identified, and the descriptors which have not yet been used (Table 7).

At this point, all other descriptors whose patterns are covered by the first level descriptors can be removed from consideration as well. The descriptors (HEIGHT=short), (HAIR=red), and (EYES=blue) need not be considered any longer. The patterns they are contained in have been dealt with by the two first level descriptors as shown in Table 8.

TABLE 7. DESCRIPTORS AND THEIR ASSOCIATED PATTERNS AFTER REMOVING FIRST-LEVEL DESCRIPTORS

Descriptor	Associated Patterns Class o	Class a
WEIGHT=heavy	4	6,10,11,12
HEIGHT=tall	4	10
HEIGHT=short		6,11,12
HAIR=red		10,11,12
HAIR=blond	4	6
EYES=brown	4	6,10,12
EYES=blue		11

TABLE 8. DESCRIPTORS AND PATTERNS AFTER REMOVAL OF COVERED PATTERNS

Descriptor	Associated Patterns Class o	Class a
WEIGHT=heavy	4	6,10,11,12
HEIGHT=tall	4	10
HAIR=blond	4	6
EYES=brown	4	6,10,12

Since all distinct descriptors have been considered, the next step is to look for second level descriptors.

Second level descriptors are formed by the conjunction (AND) of first level descriptors. A useful second level descriptor must uniquely identify patterns of the target class. Whether a descriptor pair is useful can be determined by examining the patterns covered by both descriptors under consideration. Two criteria must be met for a second level descriptor to be valid:

The intersection of the set of patterns in the target class for both descriptors must not be empty, and
The intersection of the set of patterns in all other classes must be empty.

Both first level descriptors must then be present together in at least one pattern of the desired class, and must not be present together in any patterns belonging to any other classes.

To find a valid second level descriptor, pairs of descriptors are chosen and tested. For the first pair above, which is (WEIGHT=heavy)&(HEIGHT=tall), the intersection of the set of patterns in the target class is pattern 4. The intersection of the patterns in the non-target class, however, is pattern 10, since both descriptors are present in pattern 10. This second level descriptor is not useful, since it identifies patterns in both sets. The search will then continue with other combinations of descriptors. Only one combination is of use, namely pair (HEIGHT=tall) and (HAIR=blond). These descriptors do occur in pattern 4 in the target class, and do not occur in any other patterns belonging to other classes. The second level descriptor formed from this pair, (HEIGHT=tall)&(HAIR=blond), can therefore be used:

pattern $\in$ class o if it contains (HEIGHT=tall)&(HAIR=blond)

Pattern 4 was the only uncovered pattern in the target class remaining, and is covered by this second level descriptor. At this point, then, no more patterns need to be considered, and the complete rule can be expressed (Table 9).

TABLE 9. RESULTING RULES FOR DATASET 1

IF (WEIGHT=light) or (HAIR=dark) or (HEIGHT=tall)&(HAIR=blond)

If patterns remained uncovered in the target set at this time, then other second level descriptors can be formed. If all possible pairs of second level descriptors have been tested, and none are valid second level descriptors, then the process will continue similarly, with the search for third level descriptors formed by the conjunction of three descriptors. This process will continue until partial rules are generated to uniquely identify all patterns.

To derive rules for the remaining patterns, this process can then be repeated with class a being the target class.

Experimental Results

Dataset 1 is first presented to the Pao-Hu method, resulting in the rules shown in Table 10. These rules are sufficient to classify all patterns in Dataset 1 correctly.

TABLE 10. RULES GENERATED FROM DATASET 1 BY THE PAO-HU METHOD

```
IF (WEIGHT=light) or
   (HEIGHT=tall & HAIR=blond) or
   (HEIGHT=short & HAIR=dark) or
   (HAIR=dark & EYES=blue)
THEN class=o

IF (WEIGHT=heavy & HAIR=red) or
   (WEIGHT=heavy & HEIGHT=short & HAIR=blond) or
   (WEIGHT=heavy & HEIGHT=tall & HAIR=dark & EYES=brown)
THEN class=a
```

Using Dataset 2, the Pao-Hu method induces the rules shown in TABLE 11. These rules miss discriminating on the (WEIGHT=heavy) feature for the second partial rule in class "a". This leads to a conflict in the rules generated when an example with WEIGHT=light, HEIGHT=short, HAIR=blond is presented. These examples classify as both "o" and "a". Because the Pao-Hu method attempts to generate as few rules as possible by assuming as much as it can in the absence of conflicting examples, rules for different classes may include

TABLE 11. RULES GENERATED FROM DATASET 2 BY THE PAO-HU METHOD

```
IF (WEIGHT=light) or
   (HEIGHT=tall & HAIR=blond) or
   (HEIGHT=short & HAIR=dark) or
   (HAIR=dark & EYES=blue)
THEN class=o

IF (WEIGHT=heavy & HAIR=red) or
   (HEIGHT=short & HAIR=blond) or
   (WEIGHT=heavy & HEIGHT=tall & HAIR=dark & EYES=brown)
THEN class=a
```

the same unknown patterns.

The Pao-Hu method was then applied to Dataset 3, resulting in the rules shown in Table 12. Since the example requiring the longest rule for class "a" was not used in generating the rules, it is reasonable that the method did not generate this rule. The rules produced are again sufficient to classify the patterns used, but will result in conflicts with two of the non-training patterns. The two patterns (light, short, blond, brown) and (light, short, blond, blue) are classified as belonging to both class "a" and

class "o". Again, this is because the method tried to use as few features as needed, and in the absence of a counterexample chose (HAIR=blond & HEIGHT=short) as one rule.

TABLE 12. RULES GENERATED FROM DATASET 3 BY THE PAO-HU METHOD

```
IF (WEIGHT=light) or
   (HEIGHT=tall & HAIR=blond) or
   (HAIR=dark)
THEN class=o

IF (WEIGHT=heavy & HAIR=red) or
   (HEIGHT=short & HAIR=blond)
THEN class=a
```

ANS CLASSIFIER

An area of interest in this study was the use of ANS to perform classification tasks on patterns with non-numeric features. The approach used in this study centered on feedforward networks trained with the backpropagation algorithm. Since the neural networks considered are numeric by nature, work had to be done to devise a method to convert data from a non-numeric representation to a numeric one. Previous work proposed several ways to provide numeric input. One was to present a normalized average of the ASCII codes of the letters to the network, another to select a number for each value of each feature that was far away from other numbers (0.1 for 'light', 0.9 for 'dark', for example). Each of these potential solutions had problems. The average ASCII value of a character string is not unique ('light' and 'glith' will have the same average, for example), and for a large number of values it is difficult to pick values that are 'far away' from each other without knowing all data to be considered a priori. In light of these problems and the importance of data representation to the network, encoding schemes that would allow automatic and unique representation of each input feature were examined.

BINARY ENCODING SCHEME

A solution that seemed promising was to create a bitstring

representation of each feature and present the individual bits to a network. For a given string representing a nonnumeric feature, convert each character to its ASCII code, convert the ASCII code to a binary representation, then present each bit to an input neuron in the network. Enough neurons were allocated to deal with the value having the most characters. Those values with fewer letters were padded on the right with spaces. This is referred to as the binary encoding scheme.

As an illustration, consider the feature (WEIGHT=light).

Character string:	l	i	g	h	t
	↓	↓	↓	↓	↓
ASCII code:	108	105	103	104	116
	↓	↓	↓	↓	↓
Binary:	1101100	1101001	1100111	1101000	1110100

For this feature, then, thirty-five input neurons must be allocated in the network (7 bits * 5 characters).

To allow the network to process these bitstrings, a layer of intermediate layer neurons was added. These neurons were fully connected to all input neurons for the feature, and allow the network to have an intermediate representation of the features. For the input data sets considered, the greatest number of possible values for a given feature was three (HAIR could take on the values red, dark, or blonde), so two neurons were assigned to this layer for all features (allowing them to reflect a maximum of four values). A graphical depiction of the input and classification layers for a single feature is shown (Figure 5). For this feature, seven input neurons were allocated for each of five characters, and two neurons were allocated in the classifier layer (allowing the classifier layer to reflect four different values for this feature). The neurons in the classifier layer are fully connected to the neurons in the hidden layer.

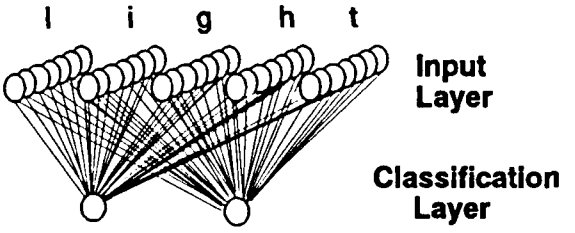

FIGURE 5. INPUT AND CLASSIFICATION LAYERS FOR ONE FEATURE

The input and classification layers are separate for different features, so the network may be scaled to allow varying numbers of features by adding or deleting these blocks of neurons. Most data sets considered in testing had three or four features per pattern. The network architecture was scaled to fit the input.

Pictured below is a graphical representation of the network architecture as applied to the problem of classifying the sample pattern set (Figure 6).

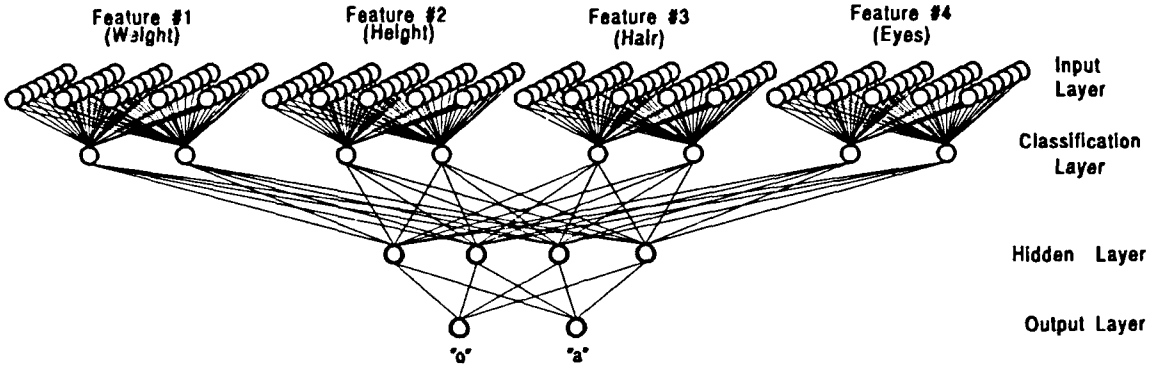

FIGURE 6. NETWORK ARCHITECTURE USING BINARY ENCODING SCHEME

As mentioned earlier, this network was trained using backpropagation of error (the delta rule). No values were specified in advance for the internal neurons in either the classification layer of the hidden layer. As a result, during the course of training the backpropagation algorithm came up with its own internal representation of the data in both the hidden and the classifier layer. For example, during one run using the data from Dataset 4, the network derived the representation for the nodes in the classifier layer shown in Table 13.

TABLE 13. REPRESENTATION OF FEATURES IN CLASSIFICATION LAYER OF BINARY ENCODED NETWORK

Feature 1 Weight		Feature 2 Height		Feature 3 Hair		Feature 4 Eyes	
light	1 0	tall	1 0	dark	0 1	brown	0 0
heavy	0 1	short	0 0	blond	1 1	blue	0 0
				red	1 0		

The values for each neuron pair were not consistent between runs, since they depend primarily on the initial random starting weights. For example, during another run, the representation in the classifier layer for the first feature was 1 and 1 for 'light', and 0 and 0 for 'heavy'.

Note that the network did not learn a representation of the

fourth feature. The network did not learn any correlation between the value of the fourth feature and the desired output, and so did not learn a representation for its values. The network decided that the value of the fourth feature was not significant in the data given.

One argument against use of this representation is that this approach uses a large amount of resources in the form of these connections to do something that is fairly simple, namely to classify a string as one of two or three classes. The net has to figure out internal representations on two levels, which is inefficient by nature of the backpropagation learning algorithm. For these reasons, a different encoding scheme was adopted for most experiments performed.

FEATURE-VALUE ENCODING SCHEME

One of the primary drawbacks of the binary encoding scheme is the large number of connections required between the input and classification layers, which resulted in a relatively large amount of computer time in order to train each network. An alternate scheme used only one neuron per possible feature value. The input patterns were preprocessed. Each input neuron was allowed to fire if the corresponding value was present in the pattern, and prevented from firing if the value was not present.

Figure 7 shows one group of input neurons, representing the possible non-numeric values of light, dark, or red, indicating the value 'dark'.

This method is easily scaled, since to reflect a new non-numeric value all that must be done is to add a neuron to the input layer.

To represent an input pattern as a vector, the possible values of each feature must be listed to determine their number. All values of the same feature are grouped together, but this is done only for convenience, and is not relevant to the representation. Shown in Table 14 are the patterns of dataset 4 and the vectors constructed from them to be presented to the network.

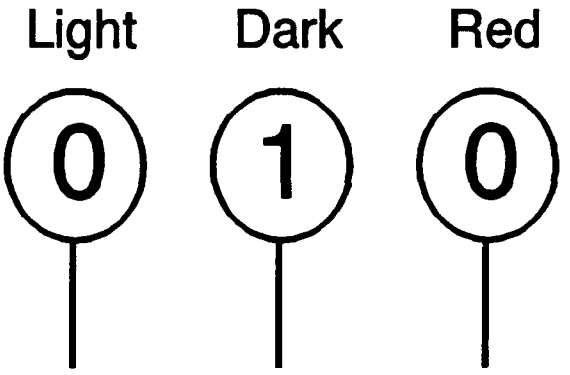

FIGURE 7. INPUT NEURON GROUP REFLECTING A VALUE OF 'DARK'

TABLE 14. VECTOR REPRESENTATION OF DATASET 4 FOR THE FEATURE-VALUE NETWORK

PATTERNS				VECTORS									
WEIGHT	HEIGHT	HAIR	EYES	light	heavy	tall	short	dark	blond	red	brown	blue	
light,	tall,	dark,	brown →	(1	0	1	0	1	0	0	1	0	)
light,	short,	dark,	brown →	(1	0	0	1	1	0	0	1	0	)
light,	short,	blond,	blue →	(1	0	0	1	0	1	0	0	1	)
heavy,	tall,	blond,	brown →	(0	1	1	0	0	1	0	1	0	)
heavy,	short,	dark,	brown →	(0	1	0	1	1	0	0	1	0	)
heavy,	short,	blond,	brown →	(0	1	0	1	0	1	0	1	0	)
light,	tall,	red,	brown →	(1	0	1	0	0	0	1	1	0	)
light,	short,	red,	blue →	(1	0	0	1	0	0	1	0	1	)
light,	short,	red,	brown →	(1	0	0	1	0	0	1	1	0	)
heavy,	tall,	red,	brown →	(0	1	1	0	0	0	1	1	0	)
heavy,	short,	red,	blue →	(0	1	0	1	0	0	1	0	1	)
heavy,	short,	red,	brown →	(0	1	0	1	0	0	1	1	0	)

The tradeoff with this method is that some time must be spent building the vectors to present them to the network. However, since this scheme contains fewer neurons (by a factor of about nine) and fewer weights (by a factor of seven), the faster training time more than makes up for the preprocessing time.

When all nine possible feature-value combinations are included, there are nine input neurons in the network. Figure 8 shows the network architecture used by the feature-value encoding scheme.

The neural network will, in general, successfully learn all those patterns presented to it. Some problems occasionally arise with local minima, but these are uncommon with the network as presented. The interest, then, is to determine how the network generalizes in response to incomplete examples. For this reason, Dataset 1 was used as a complete dataset, and datasets 2 and 3 reflect subsets of the dataset 1.

When trained on the patterns in Dataset 1, the network learned all patterns. The rules generated from this network are shown in Table 15.

When trained with all patterns from Dataset 2, the network learned to classify all patterns of Dataset 1 correctly in six out of nine training sessions. Of the remaining three, in one case the network fell into a local minimum, and for several thousand

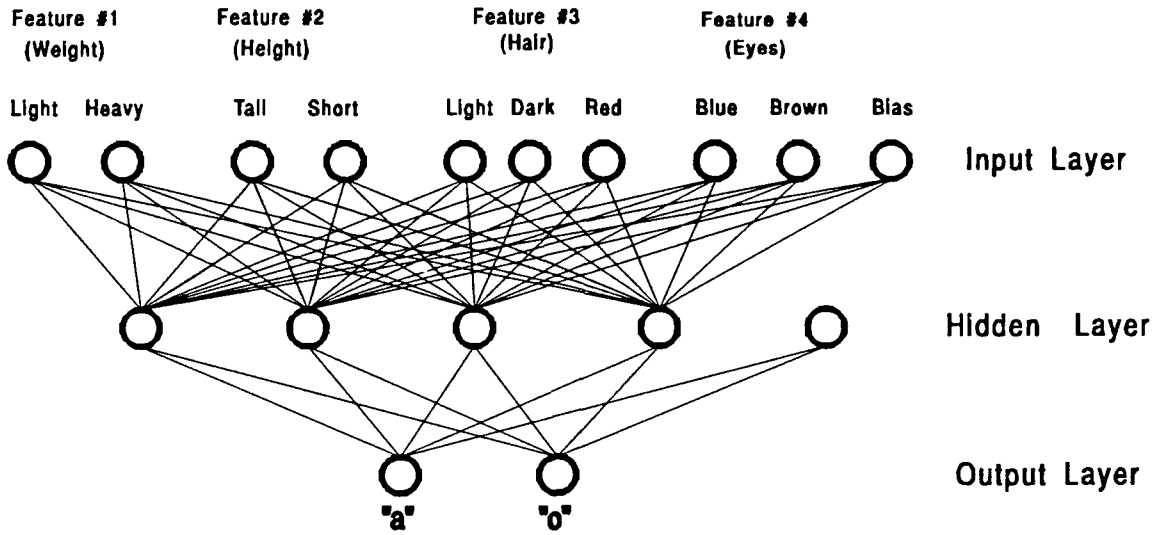

FIGURE 8. NETWORK ARCHITECTURE USING THE FEATURE-VALUE ENCODING SCHEME

TABLE 15. RULES GENERATED FROM NETWORK TRAINED ON DATASET 1

```
IF (WEIGHT=light) or
   (HEIGHT=short & HAIR=dark) or
   (HAIR=dark & EYES=blue) or
   (HEIGHT=tall & HAIR=blond)
THEN class=o


IF (WEIGHT=heavy & HAIR=red) or
   (WEIGHT=heavy & HEIGHT=short & HAIR=blond) or
   (WEIGHT=heavy & HAIR=dark & HEIGHT=tall & EYES=brown)
THEN class=a
```

iterations after learning all other patterns correctly, the network's conclusion about the membership for four patterns was still indeterminate. In the remaining two training sessions, the network incorrectly identified the pattern (light, tall, red, brown) as class "a". Inspection of the training set shows several patterns that differ from the incorrect pattern in only one feature:

(light, tall, dark, brown) of class o,
(light, tall, red, blue) of class o, and
(heavy, tall, red, brown) of class a.

Expressed as vectors, this means that

(1 0 1 0 0 0 1 1 0) (the unknown)

was most often associated with the training patterns

(1 0 1 0 1 0 0 1 0) (class o)

and

(1 0 1 0 0 0 1 0 1) (class o),

but was sometimes associated most strongly with

(0 1 1 0 0 0 1 1 0) (class a).

In these two cases, the network learned that the case where height is tall, hair is red, and eyes are brown carried more significance than the weight, which should be the primary distinguishing feature for this dataset. Since the missed example was not part of the training set, misclassification error caused by this pattern was not included in computing the overall error used by backpropagation. A representation that correctly reflected the training patterns but not the testing patterns was arrived at by the network.

For a weight set trained on Dataset 2 which correctly identified all patterns, rules were generated from the network. They are shown in Table 16. These rules successfully classify all patterns in Dataset 1.

TABLE 16. RULES GENERATED FROM NETWORK TRAINED ON DATASET 2

```
IF (WEIGHT=light) or
   (HEIGHT=short & HAIR=dark) or
   (HAIR=dark & EYES=blue) or
   (HEIGHT=tall & HAIR=blond)
THEN class=o


IF (WEIGHT=heavy & HAIR=red) or
   (WEIGHT=heavy & HEIGHT=short & HAIR=blond) or
   (WEIGHT=heavy & HAIR=dark & HEIGHT=tall & EYES=brown)
THEN class=a
```

In another example where the network was presented with a subset of Dataset 1, the network consistently found a solution where all patterns but one were identified correctly. When trained on the patterns in Dataset 3, the network consistently identified (heavy, tall, dark, brown) as of class a.

Again, in inspecting the vectors it can be seen that

(0 1 1 0 1 0 0 1 0) (the misidentified vector)

is most similar to the training vectors in three of the four features:

(0 1 0 1 1 0 0 1 0) (class o) and
(0 1 1 0 0 1 0 1 0) (class o).

The network determined the class of an unknown vector by those known vectors which were most near it.

When the network was trained on Dataset 3, it resulted in rules shown in Table 17. The rules are shorter and less specific than those of the previous two cases. As a result, the rules misclassify one pattern, the pattern with features (heavy, tall, dark, brown). This pattern is the same as that misclassified by the ID3 method.

TABLE 17. RULES GENERATED FROM NETWORK TRAINED ON DATASET 3

```
IF (WEIGHT=light) or
   (HEIGHT=tall & HAIR=blond) or
   (HAIR=dark)
THEN class=o


IF (WEIGHT=heavy & HAIR=red) or
   (WEIGHT=heavy & HEIGHT=short & HAIR=blond)
THEN class=a
```

DISCUSSION

The three methods discussed in this paper were applied to Dataset 2. The results of the Pao-Hu are shown in TABLE 18, the ID3 algorithms in Figure 9, and the results of rules generated from a neural network trained on Dataset 2 in Table 19.

While the ID3 tree correctly classifies all patterns in Dataset 1, the Pao-Hu rules misclassify (light, short, blond, blue) and (light, short, blond, brown) as both class "a" and class "o", as discussed previously.

TABLE 18. RULES FOR DATASET 2 AS GENERATED BY THE PAO-HU METHOD

```
IF (WEIGHT=light) or
   (HAIR=dark)&(EYES=blue) or
   (HAIR=dark)&(HEIGHT=short) or
   (HEIGHT=tall)&(HAIR=blond)
THEN class=o

IF (WEIGHT=heavy)&(HAIR=red) or
   (HEIGHT=short)&(HAIR=blond) or
   (WEIGHT=heavy)&(EYES=brown)&(HAIR=dark)&(HEIGHT=tall)
THEN class=a
```

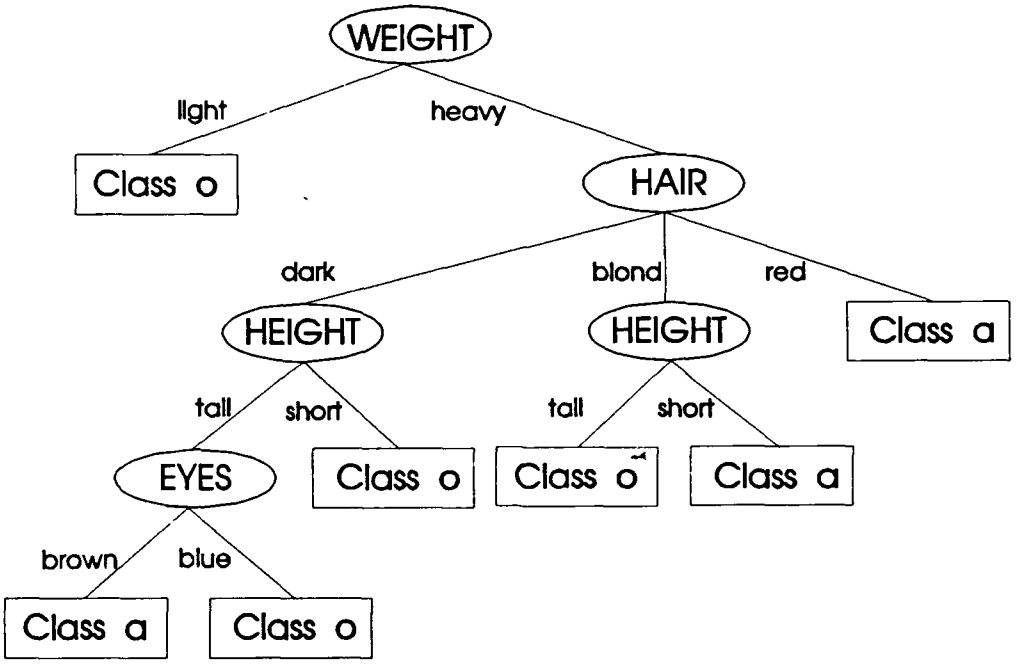

FIGURE 9. CLASSIFIER TREE GENERATED BY ID3 ON DATASET 2

The ANS used to generate rules classified all patterns successfully. The rules generated from this network also classify all patterns successfully. By retaining more conditions in the rules for class a than the Pao-Hu method, the rules do not classify any pattern as being of both classes. Note that the rules generated are identical to those generated by the Pao-Hu method on the entire Dataset 1.

The three methods were then applied to Dataset 3. The rules produced with the Pao-Hu method are shown in Table 20, the tree produced by the ID3 algorithm is shown in Figure 10, and the rules derived from a neural network trained on Dataset 3 are shown in Table 21.

TABLE 19. RULES GENERATED FROM NETWORK TRAINED ON DATASET 2

```
IF (WEIGHT=light) or
   (HEIGHT=short & HAIR=dark) or
   (HAIR=dark & EYES=blue) or
   (HEIGHT=tall & HAIR=blond)
THEN class=o


IF (WEIGHT=heavy & HAIR=red) or
   (WEIGHT=heavy & HEIGHT=short & HAIR=blond) or
   (WEIGHT=heavy & HAIR=dark & HEIGHT=tall & EYES=brown)
THEN class=a
```

TABLE 20. RULES FOR DATASET 3 GENERATED BY PAO-HU METHOD

```
IF (WEIGHT=light) or
   (HAIR=dark) or
   (HEIGHT=tall)&(HAIR=blond)
THEN class=o

IF (HAIR=blond)&(HEIGHT=short) or
   (WEIGHT=heavy)&(HAIR=red)
THEN class=a
```

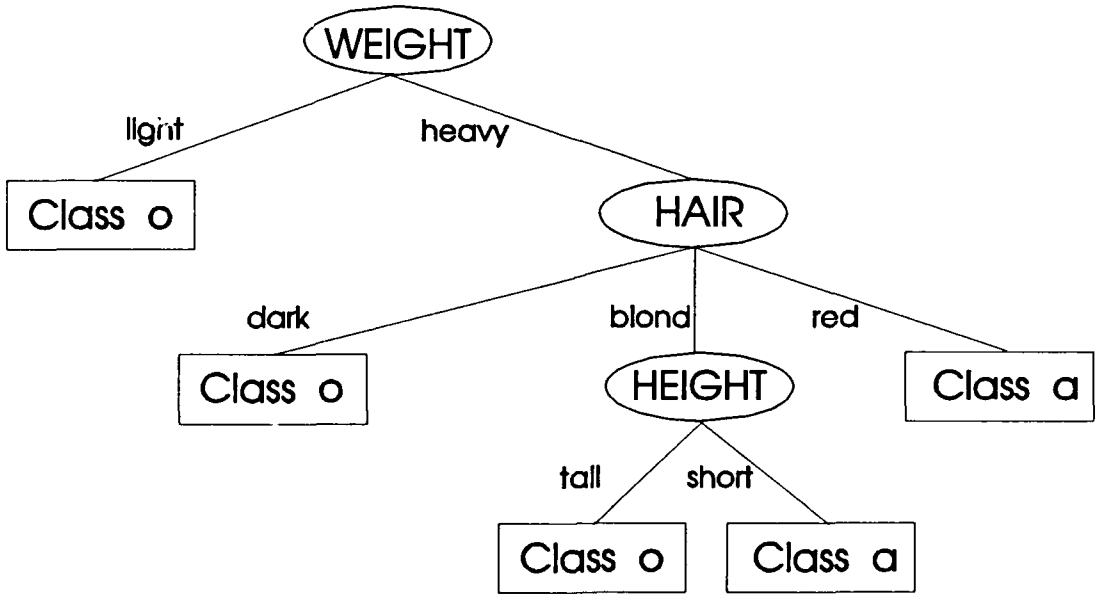

FIGURE 10. CLASSIFIER TREE GENERATED BY ID3 FROM DATASET 3

TABLE 21. RULES GENERATED FROM NETWORK TRAINED ON DATASET 3

```
IF (WEIGHT=light) or
   (HEIGHT=tall & HAIR=blond) or
   (HAIR=dark)
THEN class=o


IF (WEIGHT=heavy & HAIR=red) or
   (WEIGHT=heavy & HEIGHT=short & HAIR=blond)
THEN class=a
```

The Pao-Hu method misclassifies both (light, short, blond, brown) and (light, short, blond, blue) as both class "a" and class "o", and the ID3 algorithm misclassifies (heavy, tall, dark, brown) as of class o. Again, both mistakes relate to the nature of the methods used. The Pao-Hu chose sufficient rules on the basis of the examples it had, and ID3 chose a tree that would allow the greatest number of known patterns to be identified at each step as possible.

The ANS trained on this data classified all patterns correctly but (heavy, tall, dark, brown). This is the same pattern that the ID3 method misclassified. This similarity can probably be attributed to the fact that both methods attempt to classify patterns in a statistical manner. Both weigh the values of those features that give more information about the class by finding a correlation between an input neuron and an output neuron firing (in the network) or by finding those values which reduce the number of misidentified patterns the most at any given point (in the ID3).

CONCLUSION

The ID3 algorithm creates a very efficient decision tree for classifying patterns with nonnumeric feature values. The algorithm appears to work best when there is a large number of patterns in the data set, and when the patterns consist of several feature values.

There are two drawbacks in using the ID3 approach. The first is that there is no easy way to modify the decision tree once it has been created. Therefore, if the tree incorrectly classifies patterns as a result of not having a large enough data set, then a new one will have to be created given a larger data set. The

second drawback is that the algorithm does not have the ability to generalize. It can only create a decision tree to classify the types of patterns that were present at the time when the tree is created.

While the Pao-Hu method offers advantages on a true parallel machine, if the rules are examined on a serial machine the decision tree will allow classification in a much smaller average amount of time. This is because the Pao-Hu method considers a relatively large number of partial rules, each of which must be evaluated one after another, with no knowledge gained from evaluating one partial rule used in evaluating the next partial rule. Additionally, the emphasis in creating rules was not to minimize the number of rules but their length.

The independence of the Pao-Hu rules also gives some advantage if they are to be updated. If a new pattern is discovered that is in conflict with a partial rule, then that partial rule can be made more specific and not fire when the new pattern is presented, and the new pattern can be added into the rulebase by indicating its class and forming a partial rule of its descriptors.

A layered neural network was devised. Several methods were explored to encode the nonnumeric textual data into numeric format for input to the neural network. A method for generating rules from a backpropagation network was developed.

The generalization capabilities of the Pao-Hu and ANS classifiers were compared with each other and with the ID3 classification method. Classifiers were compared by applying all methods to several data sets and examining the similarities and differences among them. It was demonstrated that artificial neural systems can act as rule generators for expert system. The test result has shown that ANS can correctly generate and infer rules and subsequently correctly classify patterns.

REFERENCES

1. Quinlan, J. R., "Learning efficient classification procedures and their application to chess end games," in Machine Learning, eds. R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, Tioga, Palo Alto, CA, 1983.

2. Pao, Y., Adaptive Pattern Recognition and Neural Networks, Addison-Wesley Publishing, Reading, MA, 1989.

3. Allen, R.B., M.E. Riecken, Interlacing and Communicating Connectionist Agents, International Neural Network Society, Boston, pp67, September 1988.

4. Rumelhart, D., J. McClelland, Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol.1, The MIT press, 1986.

5. Gallant, S.I., Connectionist Expert System, Communications of the ACM, Vol.31 No.2, pp152-169, 1988.

DISTRIBUTION

	Copies
Chief of Naval Operations Department of the Navy	
Attn: OP-03	1
OP-033	1
OP-035	1
OP-037	1
Washington, DC 20360	
Commander Naval Sea Systems Command	
Attn: SEA-06	1
PMS-400	1
PMS-407	1
Washington, DC 20362-5101	
Office of Naval Technology	
Attn: Code 23	1
Code 213 (Siegel)	1
Code 227 (E. Wald)	1
800 No. Quincy Street Arlington, VA 22217-5000	
Office Chief of Naval Research	
Attn: Cognitive and Neural Science Division (T. McKenna)	1
800 No. Quincy Street Arlington, VA 22217-5000	
Office Chief of Naval Research	
Attn: Electronics Division (C. Lau)	1
800 No. Quincy Street Arlington, VA 22217-5000	
Director National Security Agency	
Attn: R51 (Jim Maar)	1
Ft. Meade, MD 20755-6000	
Defense Advanced Research Projects Agency	
Attn: Dso Youn	1
1400 Wilson Blvd. Roslyn, VA 22201	
Commanding Officer Naval Research Laboratory	
Attn: Code 5510 (Dr Meyrowitz)	1
4555 Overlook Avenue, SW Washington, DC 20375-5000	
National Science Foundation	
Attn: Paul Werbos Rm 1151	1
Washington, DC 20550	
Defense Technical Information Center	12
Cameron Station Alexandria, VA 22304-6145	
Library of Congress	4
Attn: Gift & Exchange Division Washington, DC 20540	

DISTRIBUTION (Cont.)

	Copies
Internal Distribution:	
A	1
B	1
B10 (Glass)	1
B10 (Priebe)	1
B10 (Reid)	1
B10 (Rogers)	1
B10 (Solka)	1
B40	1
B40 (Farsaie, Elkins)	15
C	1
D	1
D2	1
D4	1
E231	2
E232	3
F	1
F30	1
F33	1
G	1
G20	1
G30	1
G40	1
G42	1
J	1
K	1
K10	1
K50	1
L	1
L10	1
L11 (McClintock)	1
N	1
N04W	1
N10	1
N30	1
N50	1
N60	1
E61 (Valdez)	2
R	1
R04	1
R05	1
R40	1
R44	1
R44 (Szu, Telfer)	2
R46	1

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704 0188), Washington, DC 20503

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	June 1991	

4. TITLE AND SUBTITLE

Nonnumeric Pattern Classifiers

5. FUNDING NUMBERS

6. AUTHOR(S)

L. R. Elkins, A. Farsaie, and D. Valdez

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Surface Warfare Center
10901 New Hampshire Avenue
Silver Spring, MD 20903-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

NAVSWC TR 91-358

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Theoretical study was performed to determine if artificial neural systems can be used to generalize rules from examples. A method for generating rules from a multilayer network was investigated.

When the network was presented with few training patterns, the rules derived from the network classified all patterns correctly. The generalization capabilities of the Pao-Hu and neural network classifiers were compared with each other and with the ID3 method. Classifiers were compared by applying all methods to several data sets and examining the similarities and differences among them. It was demonstrated that the neural network could act as a rule generator for an expert system. The tests have shown that the network can correctly generate rules and subsequently correctly classify patterns.

14. SUBJECT TERMS

artificial neural networks, pattern classifiers, expert systems, machine learning

15. NUMBER OF PAGES

42

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR